

PPPPPPPPPPPP		AAAAAAAAAA		SSSSSSSSSSSS		CCCCCCCCCCCC		AAAAAAAAAA		LLL
PPPPPPPPPPPP		AAAAAAAAAA		SSSSSSSSSSSS		CCCCCCCCCCCC		AAAAAAAAAA		LLL
PPPPPPPPPPPP		AAAAAAAAAA		SSSSSSSSSSSS		CCCCCCCCCCCC		AAAAAAAAAA		LLL
PPP	PPP	AAA	AAA	SSS		CCC		AAA	AAA	LLL
PPP	PPP	AAA	AAA	SSS		CCC		AAA	AAA	LLL
PPP	PPP	AAA	AAA	SSS		CCC		AAA	AAA	LLL
PPP	PPP	AAA	AAA	SSS		CCC		AAA	AAA	LLL
PPP	PPP	AAA	AAA	SSS		CCC		AAA	AAA	LLL
PPP	PPP	AAA	AAA	SSS		CCC		AAA	AAA	LLL
PPPPPPPPPPPP		AAA	AAA	SSS	SSSSSSSSSS	CCC		AAA	AAA	LLL
PPPPPPPPPPPP		AAA	AAA		SSSSSSSSSS	CCC		AAA	AAA	LLL
PPPPPPPPPPPP		AAA	AAA		SSSSSSSSSS	CCC		AAA	AAA	LLL
PPP		AAAAAAAAAAAAAAAA			SSS	CCC		AAAAAAAAAAAAAAAA		LLL
PPP		AAAAAAAAAAAAAAAA			SSS	CCC		AAAAAAAAAAAAAAAA		LLL
PPP		AAAAAAAAAAAAAAAA			SSS	CCC		AAAAAAAAAAAAAAAA		LLL
PPP		AAA	AAA		SSS	CCC		AAA	AAA	LLL
PPP		AAA	AAA		SSS	CCC		AAA	AAA	LLL
PPP		AAA	AAA		SSS	CCC		AAA	AAA	LLL
PPP		AAA	AAA		SSS	CCC		AAA	AAA	LLL
PPP		AAA	AAA	SSSSSSSSSSSS		CCC		AAA	AAA	LLL
PPP		AAA	AAA	SSSSSSSSSSSS		CCCCCCCCCCCC		AAA	AAA	LLLLLLLLLLLLLLLL
PPP		AAA	AAA	SSSSSSSSSSSS		CCCCCCCCCCCC		AAA	AAA	LLLLLLLLLLLLLLLL
PPP		AAA	AAA	SSSSSSSSSSSS		CCCCCCCCCCCC		AAA	AAA	LLLLLLLLLLLLLLLL

```
PPPPPPPP      AAAAAA      SSSSSSSS      111111      000000      333333
PPPPPPPP      AAAAAA      SSSSSSSS      111111      000000      333333
PP      PP      AA      AA      SS      11      00      00      33      33
PP      PP      AA      AA      SS      11      00      00      33      33
PP      PP      AA      AA      SS      11      00      00      33      33
PP      PP      AA      AA      SS      11      00      00      33      33
PPPPPPPP      AA      AA      SSSSSS      11      00      00      33
PPPPPPPP      AA      AA      SSSSSS      11      00      00      33
PP      AAAAAAAAAA      SS      11      00      00      33
PP      AAAAAAAAAA      SS      11      00      00      33
PP      AA      AA      SS      11      00      00      33
PP      AA      AA      SS      11      00      00      33
PP      AA      AA      SSSSSSSS      111111      000000      333333
PP      AA      AA      SSSSSSSS      111111      000000      333333
                                     ....
                                     ....
                                     ....
                                     ....
```

```
LL      111111      SSSSSSSS
LL      111111      SSSSSSSS
LL      11      SS
LL      11      SS
LL      11      SS
LL      11      SS
LL      11      SSSSSS
LL      11      SSSSSS
LL      11      SS
LL      11      SS
LL      11      SS
LL      11      SS
LLLLLLLLLLLL 111111      SSSSSSSS
LLLLLLLLLLLL 111111      SSSSSSSS
```

```

0000 1 :
0000 2 :*****
0000 3 :
0000 4 :
0000 5 :  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 6 :  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 7 :  ALL RIGHTS RESERVED.
0000 8 :
0000 9 :  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 10 :  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 11 :  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 12 :  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 13 :  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 14 :  TRANSFERRED.
0000 15 :
0000 16 :  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 17 :  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 18 :  CORPORATION.
0000 19 :
0000 20 :  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 21 :  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 22 :
0000 23 :*****
0000 24 :
0000 25 :  .TITLE  PASSIO OUTPUT                      ; PASCAL RMS Linkage
0000 26 :  .ident  'V04-000'
0000 27 :
0000 28 :*****
0000 29 :*****
0000 30 :**
0000 31 :**  PASCAL RMS LINKAGE FOR VAX-11/780
0000 32 :**  =====
0000 33 :**
0000 34 :**
0000 35 :**  VERSION V1.2 -- JANUARY 1981
0000 36 :**
0000 37 :**  DEVELOPED BY:  COMPUTER SCIENCE DEPARTMENT
0000 38 :**                UNIVERSITY OF WASHINGTON
0000 39 :**                SEATTLE, WA 98195
0000 40 :**
0000 41 :**  AUTHORS:      MARK BAILEY, JOHN CHAN, HELLMUT GOLDE
0000 42 :**
0000 43 :*****
0000 44 :*****
0000 45 :
0000 46 : Modified 08Jan80: 1) Allow output of 31 character scalar values in
0000 47 :                   PASSWRITESCAL.
0000 48 :                   2) Bugfix in PASSPUTBIN. Compiler was calling PASSWRITEOK
0000 49 :                   twice under some circumstances.
0000 50 :                   Paul Hohensee
0000 51 : 05May80: Fix PASSWRITESCAL to force output even if specified
0000 52 :           field width <= 0.
0000 53 :           Paul Hohensee
0000 54 :
0000 55 : 16Oct80: Change PRN CRLF so that lines are printed:
0000 56 :           <LF> <text> <CR>
0000 57 :           Susan Azibert

```



```

0000 58 :
0000 59 :
0000 60 : 13Jan81: Change all tests of status returns from RMS to
0000 61 : BLBC R0,label and BLBS R0,label from CMPL R0,#RMS$NORMAL;
0000 62 : BNEQ label, etc.
0000 63 : Paul Hohensee
0000 64 :
0000 65 : 13Jan81: Change scalar output so element name is right truncated
0000 66 : for short field width, right justified and blank filled
0000 67 : for long field width.
0000 68 :
0000 69 : 28Aug81: Use General addressing mode. Steve Lionel.
0000 70 : *****
0000 71 : *****
0000 72 : **
0000 73 : **
0000 74 : **
0000 75 : **
0000 76 : **
0000 77 : **
0000 78 : **
0000 79 : *****
0000 80 : *****
0000 81 :
0000 82 :
0000 83 : For any file variable the following storage is assumed:
0000 84 :
0000 85 :
0000 86 : FSB:  |-----|
0000 87 : | POINTER |
0000 88 : |-----|
0000 89 : | STATUS WORD |
0000 90 : |-----|
0000 91 : | LAST |
0000 92 : |-----|
0000 93 : | LINELIMIT |
0000 94 : |-----|
0000 95 : | LINECOUNT |
0000 96 : |-----|
0000 97 : | RECORD NUMBER |
0000 98 : |-----|
0000 99 : RAB:  | 44(HEX) BYTES |
0000 100 : | |
0000 101 : | |
0000 102 : |-----|
0000 103 : FAB:  | 50(HEX) BYTES |
0000 104 : | |
0000 105 : | |
0000 106 : |-----|
0000 107 : | |
0000 108 : | |
0000 109 : NAM:  | 38(HEX) BYTES |
0000 110 : | |
0000 111 : | |
0000 112 : |-----|
0000 113 : | |
0000 114 : | |

```

NOTE: The NAM block is allocated
for the PASCAL logical files
'INPUT' and 'OUTPUT' only.

```
0000 115 : -----
0000 116 :
0000 117 : Macro options
0000 118 :
0000 119 :         .DSABL  GBL           ; no undefined references
0000 120 :         .ENABL  FPT           ; rounded arithmetic
0000 121 :
0000 122 : External references
0000 123 :
0000 124 :         .EXTRN  PASSIOERROR
0000 125 :         .EXTRN  PASSWRITEOK
0000 126 :         .EXTRN  PASSBUFFEROVER
0000 127 :         .EXTRN  PASSWriteln
0000 128 :
0000 129 :
0000 130 :         .EXTRN  FOR$CNV_OUT_D
0000 131 :         .EXTRN  FOR$CNV_OUT_E
0000 132 :         .EXTRN  FOR$CNV_OUT_F
0000 133 :         .EXTRN  FOR$CNV_OUT_I
0000 134 :         .EXTRN  FOR$CNV_OUT_O
0000 135 :         .EXTRN  FOR$CNV_OUT_Z
0000 136 :
0000 137 : Provide definitions of system values
0000 138 :
0000 139 :         $DSCDEF           ; string descriptor definitions
0000 140 :         $FABDEF
0000 141 :         $RABDEF
0000 142 :         $RMSDEF           ; for status code checking
0000 143 :
0000 144 : PASCAL compiler constants
0000 145 :
0000 146 : Note: The constants below with the names 'PASSC_XXXXX' are
0000 147 :       used in the PASCAL compiler with the names 'XXXXX'. If the
0000 148 :       values in the compiler are altered then the values below
0000 149 :       must be altered accordingly.
0000 150 :
0000 151 :       PASSC_DFLTRECSI = 257;           ; default buffer size
0000 152 :       PASSC_NIL = 0                   ; NIL pointer
0000 153 :       PASSC_TRUE = 1                  ; TRUE
0000 154 :       PASSC_FALSE = 0                 ; FALSE
0000 155 :       PASSC_NOCARR = 0                ; no carriage control
0000 156 :       PASSC_CARRIAGE = 1              ; FORTRAN carriage control
0000 157 :       PASSC_LIST = 2                  ; LIST carriage control
0000 158 :       PASSC_PRN = 3                   ; PRN carriage control
0000 159 :
0000 160 : PRN carriage control constants
0000 161 :
0000 162 :       PRN_CRLF = ^X8D01               ; PRN carriage control constant
0000 163 :                                     ; for <LF> <text> <CR>
0000 164 :       PRN_NULL = ^X0000               ; PRN carriage control constant
0000 165 :                                     ; for no carriage control
0000 166 :
0000 167 : File status block constants
0000 168 :
0000 169 :       FSB$C_BLN = ^X18                ; FSB block length
0000 170 :       FSB$V_OPEN = 5
0000 171 :       FSB$V_EOF = 1
```

00000018


```
0000 172 : FSB$V_EOLN = 2
0000 173 : FSB$V_GET = 3
0000 174 : FSB$V_TXT = 4 ; textfile flag
0000 175 : FSB$V_RDLN = 0 ; last access READLN
0000 176 : FSB$V_DIR = 6 ; direct access flag
0000 177 : FSB$V_PUT = 7
0000 178 : FSB$V_INT = 8 ; internal flag
0000 179 : FSB$V_PRMT = 9 ; prompt flag
0000 180 : FSB$V_OUTPUT = 10 ; OUTPUT file flag
0000 181 : FSB$V_ACTIN = 11 ; actual input flag
0000 182 : FSB$V_DELZ = 30 ; delete file if empty
0000 183 : FSB$V_INC = 31 ; included file flag
0000 184 : FSB$B_CC = 6 ; carriage control byte offset
0000 185 : FSB$M_OPEN = ^X0020
0000 186 : FSB$M_EOF = ^X0002
0000 187 : FSB$M_EOLN = ^X0004
0000 188 : FSB$M_GET = ^X0008
0000 189 : FSB$M_PRMT = ^X0200
0000 190 : FSB$M_PUT = ^X00000080
0000 191 : FSB$M_TXT = ^X0010
0000 192 : FSB$M_RDLN = ^X0001
0000 193 : FSB$M_DIR = ^X00000040
0000 194 : FSB$M_INT = ^X00000100
0000 195 : FSB$M_OUTPUT = ^X0400
0000 196 : FSB$M_ACTIN = ^X0800
0000 197 : FSB$M_DELZ = ^X40000000
0000 198 : FSB$M_INC = ^X80000000
0000 199 : FSB$L_CNT = 16 ; line count (textfiles)
0000 200 : FSB$L_INC = 20 ; %INCLUDE block address
0000000C 0000 201 : FSB$L_LIM = 12 ; linelimit
0000000B 0000 202 : FSB$L_LST = 8 ; last word offset
0000 203 : FSB$L_PFSB = 20 ; related file FSB for prompting
0000 204 : ; for INPUT, has address of OUTPUT FSB
0000 205 : ; for OUTPUT, has address of INPUT FSB
0000 206 : ; (shares storage with include address
0000 207 : ; and direct access record
0000 208 : ; buffer address)
0000 209 : FSB$L_REC = 20 ; record buffer address for
0000 210 : ; direct access (shares storage
0000 211 : ; with include address and related
0000 212 : ; file FSB)
0000 213 : FSB$L_STA = 4 ; status word offset
0000 214 :
0000 215 : Character constants
0000 216 :
00000020 0000 217 : TAB = ^X09
0000 218 : SPACE = ^X20
0000000C 0000 219 : DOLLAR = ^X24
0000002A 0000 220 : FORMFEED = ^XC
0000 221 : STAR = ^X2A
0000 222 : PLUS = ^X2B
0000 223 : MINUS = ^X2D
0000 224 : POINT = ^X2E
00000030 0000 225 : ZERO = ^X30
00000031 0000 226 : ONE = ^X31
0000 227 : NINE = ^X39
0000 228 : AA = ^X41
```

```
0000 229 : DD = ^X44
0000 230 : EE = ^X45
0000 231 : ZZ = ^X5A
0000 232 : UNDERSCORE = ^X5F
0000 233 : AA_SMALL = ^X61
0000 234 : ZZ_SMALL = ^X7A
0000 235 :
0000 236 :
00000000 237 : .PSECT _PASSCODE, PIC,EXE,SHR,NOWRT
0000 238 :
0000 239 : *****
0000 240 : *
0000 241 : * PASS$PUTBIN *
0000 242 : * pas$putbinary *
0000 243 : *
0000 244 : *****
0000 245 :
0000 246 : Argument offsets
0000 247 :
0000 248 : AP ; number of arguments (1)
00000004 249 : FSB_DISP = 04 ; FSB address
0000 250 :
0000 251 : .ENTRY PASS$PUTBIN,^M<R6,R7>
00000000'GF 6C FA 0002 252 : CALLG (AP),G^PASS$WRITEOK
0000 02 11 0009 253 : brb newent
0000 00C0 000B 254 : .entry pas$putbinary,^m<r6,r7>
0000 255 : newent:
0000 256 : MOVL FSB_DISP(AP),R6 ; R6 = address of FSB
0000 257 : ADDL3 R6,#FSB$C_BLN,R7 ; R7 = address of RAB
0000 258 : $PUT RAB=R7
0000 259 : BICL2 #RAB$M_TPT,RAB$L_ROP(R7); clear TPT bit
0000 260 : BLBC R0,910$ ; branch if error
0000 261 : MOV8 #RAB$C_SEQ,RAB$B_RAC(R7); make sure sequential
0000 262 : RET
0000 263 :
0000 264 : Write error
0000 265 :
0000 266 : 910$:
0000 267 : PUSH8 R0
0000 268 : MOVZBL <RAB$C_BLN+FAB$B_FNS>(R7),-(SP)
0000 269 : PUSH8 <RAB$C_BLN+FAB$L_FNA>(R7)
00000000'GF 7E 78 A7 9A 0030 270 : CALLS #3,G^PASS$IOERROR
0000 03 03 0033 271 :
0000 003A 272 :
0000003A 273 : .PSECT _PASSCODE, PIC,EXE,SHR,NOWRT
0000 003A 274 :
0000 003A 275 : *****
0000 003A 276 : *
0000 003A 277 : * PASS$PUTTXT *
0000 003A 278 : *
0000 003A 279 : *****
0000 003A 280 :
0000 003A 281 : Increments the file pointer. If the pointer is positioned at the last
0000 003A 282 : position at entry time then the buffer has overflowed.
0000 003A 283 :
0000 003A 284 : Argument offsets
0000 003A 285 :
```



```
00000004 003A 286 ;
00000004 003A 287 ;
00000004 003A 288 ;
00000000'GF 000C 003A 289 ;
52 04 AC D0 003C 290 ;
53 18 52 C1 0043 291 ;
08 A2 62 D1 004B 292 ;
00000000'GF 07 19 004F 293 ;
6C FA 0051 294 ;
62 D6 0058 295 ;
04 04 005A 296 ;
00000004 005B 297 ;
00000004 005B 298 ;
00000004 005B 299 ;
00000004 005B 300 ;
00000004 005B 301 ;
00000004 005B 302 ;
00000004 005B 303 ;
00000004 005B 304 ;
00000004 005B 305 ;
00000004 005B 306 ;
00000004 005B 307 ;
00000004 005B 308 ;
00000004 005B 309 ;
00000004 005B 310 ;
00000004 005B 311 ;
00000004 005B 312 ;
00000004 005B 313 ;
00000004 005B 314 ;
00000004 005B 315 ;
00000008 005B 316 ;
0000000C 005B 317 ;
00000010 005B 318 ;
00000010 005B 319 ;
007C 005B 320 ;
56 04 AC D0 005D 321 ;
56 DD 0061 322 ;
00000000'GF 01 FB 0063 323 ;
OC AC D5 006A 324 ;
28 15 006D 325 ;
006F 326 ;
006F 327 ;
006F 328 ;
50 08 A6 66 C3 006F 329 ;
50 OC AC D1 0074 330 ;
09 15 0078 331 ;
56 DD 007A 332 ;
00000000'GF 01 FB 007C 333 ;
0083 334 ;
OC AC D7 0083 335 ;
OC AC 20 00 B6 00 2C 0086 336 ;
00 B6 008D 337 ;
63 08 AC 01 28 008F 337 ;
66 53 D0 0094 338 ;
0097 339 ;
04 0097 340 ;
0098 341 ;

AP ; number of arguments (1)
FSB_DISP = 04 ; FSB address

.ENTRY PASS$PUTTXT,^M<R2,R3>
CALLG (AP),G^PASS$WRITEOK
MOVL FSB_DISP(AP),R2 ; R2 = address of FSB
ADDL3 R2,FSB$C_BLN,R3 ; R3 = address of RAB
CMPL (R2),FSB$L_LST(R2)
BLSS 190$ ; branch if ok
CALLG (AP),G^PASS$BUFFEROVER ; buffer overflow
190$: INCL (R2)
RET

.PSECT _PASS$CODE, PIC,EXE,SHR,NOWRT
*****
* PASS$WRITECHAR *
*****

Writes a character to the file buffer. If the field width is less
than or equal to zero then zero field width is used (ie. no output).

Argument offsets

AP ; number of arguments (4)
FSB_DISP = 04 ; FSB address
CHR_DISP = 08 ; character value (low order byte)
FLD_DISP = 12 ; field width (by value)
NOT_DISP = 16 ; (not used)

.ENTRY PASS$WRITECHAR,^M<R2,R3,R4,R5,R6>
MOVL FSB_DISP(AP),R6 ; R6 = address of FSB
PUSHL R6
CALLS #1,G^PASS$WRITEOK
TSTL FLD_DISP(AP) ; check field width
BLEQ 199$ ; exit if zero field width

Check if enough room

SUBL3 (R6),FSB$L_LST(R6),R0 ; R0 = number of bytes left
CMPL FLD_DISP(AP),R0
BLEQ 110$
PUSHL R6
CALLS #1,G^PASS$BUFFEROVER ; buffer overflow
110$: DECL FLD_DISP(AP)
MOVC5 #0,a(R6),#SPACE,FLD_DISP(AP),a(R6); blank fill

MOVC3 #1,CHR_DISP(AP),(R3) ; put character
MOVL R3,(R6) ; update file pointer
199$: RET
```



```
0098 342 :
00000098 343 : .PSECT _PASSCODE, PIC,EXE,SHR,NOWRT
0098 344 :
0098 345 : *****
0098 346 : *
0098 347 : * PASSWRITESTR *
0098 348 : *
0098 349 : *****
0098 350 :
0098 351 : Writes a string right justified with blank fill on the designated
0098 352 : file. If the field width is smaller than the string length the string
0098 353 : is truncated on the right.
0098 354 :
0098 355 : Argument offsets
0098 356 :
0098 357 :
00000004 0098 358 : AP ; number of arguments (4)
00000008 0098 359 : FSB_DISP = 04 ; FSB address
0000000C 0098 360 : STR_DISP = 08 ; string address
00000010 0098 361 : FLD_DISP = 12 ; field width (by value)
0098 362 : LEN_DISP = 16 ; string length (by value)
0098 363 :
0098 364 : .ENTRY PASSWRITESTR,^M<R2,R3,R4,R5,R7>
009A 365 : MOVL FSB_DISP(AP),R2 ; R2 = address of FSB
009E 366 : PUSH R2
00A0 367 : CALLS #1,G^PASSWRITEOK
00A7 368 : TSTL FLD_DISP(AP)
00AA 369 : BLEQ 199$ ; exit if field width <= 0
00AC 370 :
00AC 371 : Check if passing string value or address
00AC 372 :
004 10 AC D1 00AC 373 : CMPL LEN_DISP(AP),#4
006 15 00B0 374 : BLEQ 100$
007 08 AC D0 00B2 375 : MOVL STR_DISP(AP),R7 ; R7 = address of string
004 11 00B6 376 : BRB 101$
008 377 : 100$: MOVAL STR_DISP(AP),R7 ; R7 = address of string
008 378 : 101$:
008 379 : SUBL3 (R2),FSB$L LST(R2),R0
008 380 : CMPL FLD_DISP(AP),R0
009 15 00C5 381 : BLEQ 105$
009 382 : PUSH R2
00000000'GF 01 FB 00C9 383 : CALLS #1,G^PASSBUFFEROVER ; buffer overflow
00D0 384 : 105$:
00D0 385 : SUBL3 LEN_DISP(AP),FLD_DISP(AP),R4; R4 = number of bytes to pad
008 1A 00D6 386 : BGTRU 110$ ; branch if padding required
00 B2 67 0C AC 28 00D8 387 : MOVC3 FLD_DISP(AP),(R7),a(R2) ; write width characters
00D 11 00DE 388 : BRB 111$
00 B2 54 20 00 B2 00 2C 00E0 389 : 110$: MOVC5 #0,a(R2),#SPACE,R4,a(R2) ; need to blank fill R4 bytes
0063 67 10 AC 28 00E8 390 : MOVC3 ; blank fill
00ED 391 : MOVC3 LEN_DISP(AP),(R7),(R3) ; write string
004 BC 53 D0 00ED 392 : 111$: ; update pointers
00F1 393 : MOVL R3,aFSB_DISP(AP)
004 00F1 394 : 199$:
00F2 395 : RET
00F2 396 :
000000F2 397 :
000000F2 398 : .PSECT _PASSCODE, PIC,EXE,SHR,NOWRT
```

```
00F2 399 :
00F2 400 :
00F2 401 : *****
00F2 402 : * PASSWRITESCAL *
00F2 403 : *
00F2 404 : *****
00F2 405 :
00F2 406 : Write out a scalar value on the designated text file. If the field
00F2 407 : width is less than that required for the value, the value is left truncated
00F2 408 : If the field width is greater than that required for the value, the
00F2 409 : value is right justified with blank fill.
00F2 410 :
00F2 411 : Argument offsets
00F2 412 :
00F2 413 : AP ; number of arguments (4)
00000004 00F2 414 : FSB_DISP = 04 ; FSB address
00000008 00F2 415 : SCA_DISP = 08 ; scalar value (by value)
0000000C 00F2 416 : FLD_DISP = 12 ; field width (by value)
00000010 00F2 417 : NAM_DISP = 16 ; namelist address
00000014 00F2 418 : MAX_DISP = 20 ; maximal ordinal value of
00F2 419 : scalar (by value)
00F2 420 :
00F2 421 : Constants
00F2 422 :
00000020 00F2 423 : namelen = 32 ; length in bytes of one entry in
00F2 424 : name list.
00F2 425 :
00FC 00F2 426 : .ENTRY PASSWRITESCAL,^M<R2,R3,R4,R5,R6,R7>
56 04 AC DO 00F4 427 : MOVL FSB_DISP(AP),R6 ; R6 = address of FSB
00000000 GF 56 DD 00F8 428 : PUSHL R6
57 20 08 AC C5 00FA 429 : CALLS #1,G^PASSWRITEOK
57 10 AC C0 0101 430 : MULL3 SCA_DISP(AP),#namelen,R7
0106 431 : ADDL2 NAM_DISP(AP),R7 ; R7 = scalar name address
010A 432 :
010A 433 : Calculate scalar name length and check for bounds
010A 434 :
08 AC D5 010A 435 : TSTL SCA_DISP(AP)
14 AC 08 AC D1 010D 436 : BLSS 900$
21 14 0114 437 : CMPL SCA_DISP(AP),MAX_DISP(AP)
67 20 20 3A 0116 438 : BGTR 900$
51 20 50 C3 011A 439 : LOCC #SPACE,#namelen,(R7)
011E 440 : SUBL3 R0,#namelen,R1
011E 441 :
011E 442 : Call PASSWRITESTR to actually write the value to the buffer
011E 443 :
04 51 DD 011E 444 : PUSHL R1 ; pass name length
0C AC DD 0120 445 : PUSHL FLD_DISP(AP) ; pass field width
04 51 D1 0123 446 : CMPL R1,#4 ; pass by value or reference
04 15 0126 447 : BLEQ 110$
57 DD 0128 448 : PUSHL R7 ; by reference
02 11 012A 449 : BRB 111$
67 DD 012C 450 110$: :
012E 451 : PUSHL (R7) ; by value
FF62 CF 04 AC DD 012E 452 111$: :
04 FB 0131 453 : PUSHL FSB_DISP(AP)
04 0136 454 : CALLS #4,PASSWRITESTR
455 : RET
```



```

0137 456 :
0137 457 900$:
7E 83A4 8F 3C 0137 458 MOVZWL #^X83A4,-(SP)
7E 0090 C6 9A 0137 459 MOVZBL <FSB$C_BLN+RAB$C_BLN+FAB$B_FNS>(R6),-(SP)
00000000'GF 03 DD 0141 460 PUSHL <FSB$C_BLN+RAB$C_BLN+FAB$B_FNA>(R6)
0145 461 CALLS #3,G^PASSIOERROR
014C 462 :
014C 463 :
0000014C 464 .PSECT _PASSCODE, PIC,EXE,SHR,NOWRT
014C 465 :
014C 466 *****
014C 467 *
014C 468 * PASSWRITEINT *
014C 469 *
014C 470 *****
014C 471 :
014C 472 Writes an integer right justified in the designated field width.
014C 473 If the field width is less than the minimum the minimum field width is
014C 474 used. If the integer won't fit in the designated field width then the
014C 475 amount needed is used. If the field width used will overflow the buffer
014C 476 a runtime error occurs.
014C 477 :
014C 478 Argument offsets
014C 479 :
014C 480 AP ; number of arguments (4)
00000004 014C 481 FSB_DISP = 04 ; FSB address
00000008 014C 482 INT_DISP = 08 ; integer value
0000000C 014C 483 FLD_DISP = 12 ; field width (by value)
00000010 014C 484 NOT_DISP = 16 ; (not used)
014C 485 :
014C 486 Other constants
014C 487 :
00000001 014C 488 IMINP = 1 ; minimum field width for
014C 489 ; positive integers
00000002 014C 490 IMINN = 2 ; minimum field width for
014C 491 ; negative integers
00000014 014C 492 IMAX = 20 ; maximum field width needed
014C 493 ; for integers
014C 494 :
03FC 014C 495 .ENTRY PASSWRITEINT,M<R2,R3,R4,R5,R6,R7,R8,R9>
56 04 AC D0 014E 496 MOVL FSB_DISP(AP),R6 ; R6 = address of FSB
00000000'GF 01 FB 0152 497 PUSHL R6
0154 498 CALLS #1,G^PASSWRITEOK
015B 499 :
015B 500 Make room for descriptor on stack
015B 501 :
5E 08 C2 015B 502 SUBL2 #DSC$C_S_BLN,SP
58 5E D0 015E 503 MOVL SP,R8 ; R8 = address of descriptor
0161 504 :
0161 505 Check for minimum field width (1 for positive, 2 for negative)
0161 506 :
53 0C AC D0 0161 507 MOVL FLD_DISP(AP),R3 ; R3 = field width
08 AC D5 0165 508 TSTL INT_DISP(AP) ; test sign of value
0A 19 0168 509 BLSS 110$
016A 510 :
53 01 D1 016A 511 CMPL #IMINP,R3 ; positive value
0D 15 016D 512 BLEQ 120$ ; use at least minimum
```



```
53 01 D0 016F 513      MOVL    #IMINP,R3
    08 11 0172 514      BRB     120$
    02 D1 0174 515      110$:    ; negative value
    03 15 0177 516      CMPL    #IMINN,R3      ; use at least minimum
    02 D0 0179 517      BLEQ    120$
    01 017C 518      MOVL    #IMINN,R3
    01 017C 519      120$:    ; R3 = field width
    01 017C 520      :
    01 017C 521      : Convert number to character string
    01 017C 522      :
57 08 A6 66 C3 017C 523      SUBL3   (R6),FSB$L_LST(R6),R7      ; R7 = number of bytes left in line
    57 53 D1 0181 524      CMPL    R3,R7
    09 15 0184 525      BLEQ    125$
    56 DD 0186 526      PUSHL   R6
00000000'GF 01 FB 0188 527      CALLS  #1,G^PAS$BUFFEROVER      ; buffer overflow
    01 018F 528      125$:
    68 53 B0 018F 529      MOVW    R3,DSC$W_LENGTH(R8)      ; pass field width
    04 A8 66 D0 0192 530      MOVL   (R6),DSC$A_POINTER(R8)      ; pass buffer address
    58 DD 0196 531      PUSHL   R8      ; pass descriptor address
    08 AC DD 0198 532      PUSHL   INT_DISP(AP)
00000000'GF 02 FB 019B 533      CALLS  #2,G^FOR$CNV_OUT_I
    05 50 E9 01A2 534      BLBC    R0,130$
    66 53 C0 01A5 535      ADDL2   R3,(R6)      ; update file pointer
    36 11 01A8 536      BRB     199$      ; exit, conversion succeeded
    01AA 537      :
    01AA 538      : Bad conversion; use a larger buffer and try again
    01AA 539      :
    01AA 540      130$:
    68 14 B0 01AA 541      MOVW    #IMAX,DSC$W_LENGTH(R8)      ; pass buffer length
    SE 14 C2 01AD 542      SUBL2   #IMAX,SP      ; make room for buffer on stack
    59 5E D0 01B0 543      MOVL    SP,R9
    04 A8 59 D0 01B3 544      MOVL   R9,DSC$A_POINTER(R8)      ; pass buffer address
    58 DD 01B7 545      PUSHL   R8      ; pass descriptor address
    08 AC DD 01B9 546      PUSHL   INT_DISP(AP)
00000000'GF 02 FB 01BC 547      CALLS  #2,G^FOR$CNV_OUT_I
    1B 50 E9 01C3 548      BLBC    R0,910$
    69 14 20 3B 01C6 549      SKPC   #SPACE,#IMAX,(R9)      ; skip leading spaces
    01CA 550      ; R0 = number of remaining
    01CA 551      ; characters
    01CA 552      ; R1 = address of remaining
    01CA 553      ; characters
    57 50 D1 01CA 554      CMPL    R0,R7      ; check if enough room
    09 15 01CD 555      BLEQ    140$
    56 DD 01CF 556      PUSHL   R6
00000000'GF 01 FB 01D1 557      CALLS  #1,G^PAS$BUFFEROVER      ; buffer overflow
    01D8 558      140$:
    00 B6 61 50 28 01D8 559      MOVW   R0,(R1),@ (R6)      ; move string to output buffer
    66 53 D0 01DD 560      MOVL    R3,(R6)      ; update file pointer
    04 01E0 561      199$:
    01E0 562      RET
    01E1 563      :
    01E1 564      : Output conversion error
    01E1 565      :
    01E1 566      910$:
7E 83A4 8F 3C 01E1 567      MOVZWL  #^X83A4,-(SP)
7E 0090 C6 9A 01E6 568      MOVZBL  <FSB$C_BLN+RAB$C_BLN+FAB$B_FNS>(R6),-(SP)
    0088 C6 DD 01EB 569      PUSHL  <FSB$C_BLN+RAB$C_BLN+FAB$L_FNA>(R6)
```

```
00000000'GF 03 FB 01EF 570 CALLS #3,G^PASSIOERROR
                01F6 571
                01F6 572
0000 01F6 573 .PSECT _PASSCODE, PIC,EXE,SHR,NOWRT
                01F6 574
                01F6 575 *****
                01F6 576 * PASSWRITEDOUBE *
                01F6 577 *
                01F6 578 *
                01F6 579 *****
                01F6 580
                01F6 581 Write out a double precision number in 'E' format. A minimum
                01F6 582 field width of FMIN is used.
                01F6 583
                01F6 584 Argument offsets
                01F6 585
                01F6 586 AP ; number of arguments (4)
00000004 01F6 587 FSB_DISP = 04 ; FSB address
00000008 01F6 588 DOB_DISP = 08 ; double number (by reference)
0000000C 01F6 589 FLD_DISP = 12 ; field width (by value)
                01F6 590
                01F6 591 Other constants
                01F6 592
00000008 01F6 593 FMIN = 8 ; minimum field width
                01F6 594
                003C 01F6 595 .ENTRY PASSWRITEDOUBE,^M<R2,R3,R4,R5>
                01F8 596
                01F8 597 Make room for descriptor and double precission value on stack
                01F8 598
                01F8 599 SUBL2 #<DSC$C_S_BLN+8>,SP
08 A1 5E 10 C2 01F8 600 MOVL SP,R1 ; R1 = descriptor address
        51 5E D0 01F8 601 MOVD @DOB_DISP(AP),DSC$C_S_BLN(R1); put value on stack
        08 BC 70 01FE 601 MOVL #1,R5 ; set flag
        55 01 D0 0203 602 BRW PASSWREALE ; jump to common code
        0010 31 0206 603
                0209 604
                0209 605 *****
                0209 606 * PASSWRITEREAL *
                0209 607 *
                0209 608 *
                0209 609 *****
                0209 610
                0209 611 Write a real number in 'E' format. A minimum field width of EMIN is
                0209 612 used.
                0209 613
                0209 614 Argument offsets
                0209 615
                0209 616 AP ; number of arguments (4)
00000004 0209 617 FSB_DISP = 04 ; FSB address
00000008 0209 618 REL_DISP = 08 ; real number (by value)
0000000C 0209 619 FLD_DISP = 12 ; field width (by value)
00000010 0209 620 NOT_DISP = 16 ; (not used)
                0209 621
                0209 622 Other constants
                0209 623
00000008 0209 624 EMIN = 08 ; minimum field width
                0209 625
                003C 0209 626 .ENTRY PASSWRITEREAL,^M<R2,R3,R4,R5>
```

Address	Hex	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	Op418	Op419	Op420	Op421	Op422	Op423	Op424	Op425	Op426	Op427	Op428	Op429	Op430	Op431	Op432	Op433	Op434	Op435	Op436	Op437	Op438	Op439	Op440	Op441	Op442	Op443	Op444	Op445	Op446	Op447	Op448	Op449	Op450	Op451	Op452	Op453	Op454	Op455	Op456	Op457	Op458	Op459	Op460	Op461	Op462	Op463	Op464	Op46
---------	-----	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	------


```

7E 83A4 8F 3C 0275 684 ;
7E 0090 C2 9A 0275 685 ; 910$:
00000000 GF 03 FB 0275 686 MOVZWL #^X83A4,-(SP)
027A 687 MOVZBL <FSB$C_BLN+RAB$_BLN+FAB$B_FNS>(R2),-(SP)
027F 688 PUSHL <FSB$C_BLN+RAB$_BLN+FAB$L_FNA>(R2)
0283 689 CALLS #3,G^PASSIOERROR
028A 690 ;
028A 691 ;
0000 028A 692 .PSECT _PASSCODE, PIC,EXE,SHR,NOWRT
028A 693 ;
028A 694 *****
028A 695 *
028A 696 * PASSWRITEDOUBF *
028A 697 *
028A 698 *****
028A 699 ;
028A 700 Writes out a double number in fixed format.
028A 701 ;
028A 702 Argument offsets
028A 703 ;
028A 704 AP ; number of arguments (4)
00000004 028A 705 FSB_DISP = 04 ; FSB address
00000008 028A 706 DOB_DISP = 08 ; double value (by reference)
0000000C 028A 707 FLD_DISP = 12 ; field width (by value)
00000010 028A 708 DIG_DISP = 16 ; digits to right of decimal
028A 709 ; point (by value)
028A 710 ;
028A 711 Other constants
028A 712 ;
00000003 028A 713 FMIN = 3 ; minimum field width
0000002A 028A 714 FMAX = 42 ; maximum field width
028A 715 ;
007C 028A 716 .ENTRY PASSWRITEDOUBF,^M<R2,R3,R4,R5,R6>
028C 717 ;
028C 718 Make room for descriptor and double precision value on stack
028C 719 ;
028C 720 SUBL2 #<DSC$_S_BLN+8>,SP
028F 721 MOVL SP,R1 ; R1 = address of descriptor
08 A1 08 BC 70 0292 722 MOVD @DOB_DISP(AP),DSC$_S_BLN(R1); put value on stack
000D 31 0297 723 BRW PASSWREALF ; jump to common code
029A 724 ;
029A 725 *****
029A 726 *
029A 727 * PASSWRITEREALF *
029A 728 *
029A 729 *****
029A 730 ;
029A 731 Writes out a real number in fixed format.
029A 732 ;
029A 733 Argument offsets
029A 734 ;
029A 735 AP ; number of arguments (4)
00000004 029A 736 FSB_DISP = 4 ; FSB address
00000008 029A 737 REL_DISP = 08 ; real number (by value)
0000000C 029A 738 FLD_DISP = 12 ; field width
00000010 029A 739 DIG_DISP = 16 ; digits to right of decimal
029A 740 ; point (by value)
```

```

029A 741 :
029A 742 : Other constants
029A 743 :
029A 744 : FMIN = 3 : minimum field width
029A 745 : FMAX = 42 : maximum field width
029A 746 : : (sign + point + 40)
029A 747 :
007C 029A 748 : .ENTRY PASS$WRITEREALF,^M<R2,R3,R4,R5,R6>
029C 749 :
029C 750 : Make room for descriptor and double precision value on stack
029C 751 :
029C 752 :
029C 753 :
029F 754 :
02A2 755 :
02A7 756 :
02A7 757 :
02A7 758 : The code below is common to both PASS$WRITEREALF and PASS$WRITEDOUBF.
02A7 759 : After the double precision value is placed on the stack there is no
02A7 760 : difference in the way the values are converted.
02A7 761 :
02A7 762 :
02A7 763 :
02A7 764 : Check field widths and adjust if necessary
02A7 765 :
02A7 766 :
02A7 767 : PASS$WREALF:
02AB 768 : MOVL FSB_DISP(AP),R6 : R6 = address of FSB
02AD 769 : PUSHL R6
02B4 770 : CALLS #1,G^PASS$WRITEOK
02B8 771 : MOVL FLD_DISP(AP),R3 : R3 = field width (p)
02BB 772 : TSTD DSC$C_S_BLN(R1) : check if positive
02BD 773 : BLSJ 105$
02C1 774 : MOVL #SPACE,2(R6) : if positive force blank
02C4 775 : PUSHL FSB_DISP(AP)
02C9 776 : CALLS #1,PASS$PUTTXT : write out blank
02CB 777 : DECL R3 : adjust field width by 1
02CB 778 : 105$: MOVL DIG_DISP(AP),R4 : R4 = digits to right (q)
02CF 779 : TSTL R4 : q > 0?
02D1 780 : BGEQ 110$
02D3 781 : CLRL R4 : q := 0
02D5 782 : 110$:
02D5 783 : SUBL3 R4,R3,R0
02D9 784 : CMPL R0,#2 : p > q+2?
02DC 785 : BGEQ 120$
02DE 786 : ADDL3 R4,#2,R3 : p := q+2
02E2 787 :
02E2 788 : Set up descriptor and call conversion routine
02E2 789 :
02E2 790 :
02E2 791 : 120$:
02E2 792 : SUBL3 (R6),FSB$L_LST(R6),R5 : R5 = number of bytes left in line
02E7 793 : CMPL R3,R5
02EA 794 : BLEQ 125$
02EC 795 : PUSHL R6
02EE 796 : CALLS #1,G^PASS$BUFFEROVER : buffer overflow
02F5 797 : 125$:
02F5 798 : MOVW R3,DSC$W_LENGTH(R1)

```

```
04 A1 66 D0 02F8 798
      00 DD 02FC 799
      54 DD 02FE 800
      51 DD 0300 801
00000000'GF 08 A1 DF 0302 802
      04 FB 0305 803
      05 50 E9 030C 804
      66 53 C0 030F 805
      3E 11 0312 806
           0314 807
           0314 808
           0314 809
           0314 810
53 51 5E D0 0314 811
      54 2A C1 0317 812
      5E 53 C2 031B 813
      61 53 B0 031E 814
04 A1 5E D0 0321 815
      00 DD 0325 816
      54 DD 0327 817
      51 DD 0329 818
00000000'GF 08 A1 DF 032B 819
      04 FB 032E 820
      1B 50 E9 0335 821
      6E 53 20 3B 0338 822
      55 50 D1 033C 823
      09 15 033F 824
      56 C0 0341 825
00000000'GF 01 FB 0343 826
      034A 827
00 B6 61 50 28 034A 828
      66 53 D0 034F 829
           0352 830
           04 0352 831
           0353 832
           0353 833
           0353 834
           0353 835
7E 83A4 8F 3C 0353 836
7E 0090 C6 9A 0358 837
      0088 C6 DD 035D 838
00000000'GF 03 FB 0361 839
      0000 0368 840
           0368 841
           0368 842
           0368 843
           0368 844
           0368 845
           0368 846
           0368 847
           0368 848
           0368 849
           0368 850
           0368 851
           0368 852
           0368 853
00000004 0368 854
```

```
MOVL (R6),DSC$A_POINTER(R1)
PUSHL #0 ; scale factor
PUSHL R4 ; digits in fraction
PUSHL R1 ; string descriptor
PUSHAL DSC$C_5,BLN(R1) ; value
CALLS #4,G^FOR$CNV_OUT_F
BLBC R0,130$
ADDL2 R3,(R6) ; update the file pointer
BRB 199$

: Bad conversion; use a buffer of subfield+overflowsize and try again
130$:
MOVL SP,R1 ; R1 = descriptor address
ADDL3 #FMAX,R4,R3 ; R3 = new buffer size
SUBL2 R3,SP ; make room on stack
MOVH R3,DSC$W_LENGTH(R1)
MOVL SP,DSC$A_POINTER(R1) ; buffer address
PUSHL #0
PUSHL R4 ; digits in fraction
PUSHL R1 ; descriptor address
PUSHAL DSC$C_5,BLN(R1) ; value address
CALLS #4,G^FOR$CNV_OUT_F
BLBC R0,910$
SKPC #SPACE,R3,(SP) ; skip leading blanks
CMPL R0,R5
BLEQ 140$
PUSHL R6
CALLS #1,G^PASS$BUFFEROVER ; buffer overflow

140$:
MOVC3 R0,(R1),@ (R6) ; store string
MOVL R3,(R6) ; update file pointer

199$:
RET

: Output conversion error
910$:
MOVZWL #^X83A4,-(SP)
MOVZBL <FSB$C_BLN+RAB$C_BLN+RAB$B_FNS>(R6),-(SP)
PUSHL <FSB$C_BLN+RAB$C_BLN+RAB$B_FNA>(R6)
CALLS #3,G^PASS$IOERROR
.PSECT _PASS$CODE, PIC,EXE,SHR,NOWRT

*****
* PASS$WRITEHEX *
*****

Writes out a longword in hexadecimal form. Leading zeros are printed
up to eight places.

Argument offsets
AP ; number of arguments (4)
FSB_DISP = 04 ; FSB address
```



```
00000008 0368 855 VAL_DISP = 08 ; value to be printed
0000000C 0368 856 FLD_DISP = 12 ; field width (by value)
00000010 0368 857 NOT_DISP = 16 ; (not used)
0368 858
0368 859 : Other constants
0368 860
00000008 0368 861 HMAX = 8 ; maximum zero fill field width
00000010 0368 862 OVERFLOWSIZE = 16 ; maximum overflow buffer size
0368 863
00FC 0368 864 .ENTRY PASSWRITEHEX,^M<R2,R3,R4,R5,R6,R7>
56 04 AC D0 036A 865 MOVL FSB_DISP(AP),R6 ; R6 = address of FSB
56 DD 036E 866 PUSHL R6
000000C7'GF 01 FB 0370 867 CALLS #1,G^PASSWRITEOK
OC AC D5 0377 868 TSTL FLD_DISP(AP)
65 15 037A 869 BLEQ $40 ; exit if field width <= 0
037C 870
037C 871 : Make room for descriptor on stack
037C 872
5E 08 C2 037C 873 SUBL2 #DSC$C_S_BLN,SP
52 5E D0 037F 874 MOVL SP,R2 ; R2 = address of descriptor
62 OC AC B0 0382 875 MOVW FLD_DISP(AP),DSC$W_LENGTH(R2); store length
04 A2 66 D0 0386 876 MOVL (R6),DSC$A_POINTER(R2) ; store buffer address
52 DD 038A 877 PUSHL R2
08 AC DD 038C 878 PUSHL VAL_DISP(AP)
00000000'GF 02 FB 038F 879 CALLS #2,G^FOR$CNV_OUT_2
66 OC AC C0 0396 880 ADDL2 FLD_DISP(AP),(R6)
04 B2 2A 91 039A 881 CMPB #STAR,@DSC$A_POINTER(R2); test overflow
03 13 039E 882 BEQL $25
0035 31 03A0 883 BRW $35
03A3 884 $25:
66 OC AC C2 03A3 885 SUBL2 FLD_DISP(AP),(R6) ; restore pointer
62 10 B0 03A7 886 MOVW #OVERFLOWSIZE,DSC$W_LENGTH(R2)
5E 10 C2 03AA 887 SUBL2 #OVERFLOWSIZE,SP
57 5E D0 03AD 888 MOVL SP,R7
04 A2 57 D0 03B0 889 MOVL R7,DSC$A_POINTER(R2)
52 DD 03B4 890 PUSHL R2
08 AC DD 03B6 891 PUSHL VAL_DISP(AP)
00000000'GF 02 FB 03B9 892 CALLS #2,G^FOR$CNV_OUT_2
67 10 20 3B 03C0 893 SKPC #SPACE,#OVERFLOWSIZE,(R7); skip blanks
54 50 OC AC C3 03C4 894 SUBL3 FLD_DISP(AP),R0,R4
51 54 C0 03C9 895 ADDL2 R4,R1
00 B6 61 OC AC 28 03CC 896 MOVCL FLD_DISP(AP),(R1),@ (R6) ; deposit string
66 OC AC C0 03D2 897 ADDL2 FLD_DISP(AP),(R6) ; fix up pointer
09 11 03D6 898 BRB $40
54 08 CE 03D8 899 $35:
000004C6'EF 16 03DB 900 MNEGL #HMAX,R4
04 03E1 901 JSB ZERO_FILL_R3
03E2 902 $40:
03E2 903 RET
03E2 904
0000 03E2 905 .PSECT _PASSCODE, PIC,EXE,SHR,NOWRT
03E2 906
03E2 907
03E2 908
03E2 909
03E2 910
03E2 911 *****
* PASSWRITEHEXD *
*****
```

```
03E2 912 :  
03E2 913 : Write out a double precision value (quadword) in hexadecimal form.  
03E2 914 : Leading zeros up to 16 places are printed  
03E2 915 :  
03E2 916 : Argument offsets  
03E2 917 :  
03E2 918 : AP ; number of arguments (4)  
00000004 03E2 919 : FSB_DISP = 04 ; FSB address  
00000008 03E2 920 : VAR_DISP = 08 ; value address  
0000000C 03E2 921 : FLD_DISP = 12 ; field width by value  
00000010 03E2 922 : NOT_DISP = 16 ; (not used)  
03E2 923 :  
03E2 924 : Other constants  
03E2 925 :  
00000008 03E2 926 : HMAX = 8 ; maximum field for leading zeros  
0000 03E2 927 : .ENTRY PASSWRITEHEXD,^M<>  
50 0C AC 08 C3 03E4 928 : SUBL3 #HMAX,FLD_DISP(AP),R0 ; R0 = field width low bytes  
06 14 03E9 929 : BGTR 110$  
50 0C AC D0 03EB 930 : MOVL FLD_DISP(AP),R0  
16 11 03EF 931 : BRB 111$  
03F1 932 :  
03F1 933 : Print low order longword  
03F1 934 :  
03F1 935 : 110$:  
00 DD 03F1 936 : PUSHL #0  
50 DD 03F3 937 : PUSHL R0  
04 AC C1 03F5 938 : ADDL3 VAR_DISP(AP),#4,R0  
60 DD 03FA 939 : PUSHL (R0) ; low order longword  
04 AC DD 03FC 940 : PUSHL FSB_DISP(AP)  
FF64 CF 04 FB 03FF 941 : CALLS #4,PASSWRITEHEX  
50 08 D0 0404 942 : MOVL #HMAX,R0 ; field width high bytes  
0407 943 :  
0407 944 : Print R0 digits of high order longword  
0407 945 :  
0407 946 : 111$:  
00 DD 0407 947 : PUSHL #0  
50 DD 0409 948 : PUSHL R0  
08 BC DD 040B 949 : PUSHL @VAR_DISP(AP)  
04 AC DD 040E 950 : PUSHL FSB_DISP(AP)  
FF52 CF 04 FB 0411 951 : CALLS #4,PASSWRITEHEX  
04 0416 952 : RET  
0417 953 :  
0417 954 :  
0000 0417 955 : .PSECT _PASSCODE, PIC,EXE,SHR,NOWRT  
0417 956 :  
0417 957 : *****  
0417 958 : * PASSWRITEOCT *  
0417 959 : *  
0417 960 : *****  
0417 961 :  
0417 962 :  
0417 963 : Argument offsets  
0417 964 :  
00000004 0417 965 : AP ; number of arguments (4)  
00000008 0417 966 : FSB_DISP = 04 ; FSB address  
0000000C 0417 967 : VAL_DISP = 08 ; value to be printed  
0417 968 : FLD_DISP = 12 ; field width
```

```
00000010 0417 969      NOT_DISP = 16                ; (not used)
          0417 970      ;
          0417 971      ; Other constants
          0417 972      ;
00000008 0417 973      OMAX = 11                    ; maximum field for leading zeros
00000014 0417 974      OVERFLOWSIZE = 20            ; overflow buffer size
          0417 975      ;
          00FC 0417 976      .ENTRY PASS$WRITEOCT, ^M<R2,R3,R4,R5,R6,R7>
56 04 AC D0 0419 977      MOVL FSB_DISP(AP),R6      ; R6 = address of FSB
          56 DD 041D 978      PUSHL R6
00000000'GF 01 FB 041F 979      CALLS #1,G^PASS$WRITEOK
          OC AC D5 0426 980      TSTL FLD_DISP(AP)
          65 15 0429 981      BLEQ $43              ; exit if field width <= 0
          042B 982      ;
          042B 983      ; Make room for descriptor on stack
          042B 984      ;
          5E 08 C2 042B 985      SUBL2 #DSC$C_S_BLN,SP
          52 5E D0 042E 986      MOVL SP,R2
62 0C AC B0 0431 987      MOVW FLD_DISP(AP),DSC$W_LENGTH(R2); store length
04 A2 66 D0 0435 988      MOVL (R6),DSC$A_POINTER(R2) ; store buffer address
          52 DD 0439 989      PUSHL R2
          08 AC DD 043B 990      PUSHL VAL_DISP(AP)
00000000'GF 02 FB 043E 991      CALLS #2,G^FOR$CNV_OUT_0
          66 OC AC C0 0445 992      ADDL2 FLD_DISP(AP),(R6)
04 B2 2A 91 0449 993      CMPB #STAR,@DSC$A_POINTER(R2); test overflow
          03 13 044D 994      BEQL $55
          0035 31 044F 995      BRW $65
          0452 996 $55:
66 0C AC C2 0452 997      SUBL2 FLD_DISP(AP),(R6)      ; restore pointer
          62 14 B0 0456 998      MOVW #OVERFLOWSIZE,DSC$W_LENGTH(R2)
          5E 14 C2 0459 999      SUBL2 #OVERFLOWSIZE,SP
          57 5E D0 045C 1000      MOVL SP,R7
04 A2 57 D0 045F 1001      MOVL R7,DSC$A_POINTER(R2)
          52 DD 0463 1002      PUSHL R2
          08 AC DD 0465 1003      PUSHL VAL_DISP(AP)
00000000'GF 02 FB 0468 1004      CALLS #2,G^FOR$CNV_OUT_0
          67 14 20 3B 046F 1005      SKPC #SPACE,#OVERFLOWSIZE,(R7); skip blanks
54 50 OC AC C3 0473 1006      SUBL3 FLD_DISP(AP),R0,R4
          51 54 C0 0478 1007      ADDL2 R4,R1
00 B6 61 OC AC 28 047B 1008      MOVCL3 FLD_DISP(AP),(R1),@ (R6) ; deposit string
          66 OC AC C0 0481 1009      ADDL2 FLD_DISP(AP),(R6) ; fix up pointer
          09 11 0485 1010      BRB $43
          54 0B CE 0487 1011 $65:
000004C6'EF 16 048A 1013 $43:
          04 0490 1014      RET
          0491 1015      ;
          0491 1016      ;
0000 0491 1017      .PSECT _PASS$CODE,                PIC,EXE,SHR,NOWRT
          0491 1018      ;
          0491 1019      ; *****
          0491 1020      ; * PASS$WRITEOCTD *
          0491 1021      ; *
          0491 1022      ; *****
          0491 1023      ;
          0491 1024      ;
          0491 1025      ; Write out a double precision value (quadword) in octal format.
```



```
0491 1026 : Leading zeros up to twenty-two places are printed.
0491 1027 :
0491 1028 : Argument offsets
0491 1029 :
0491 1030 : AP : number of arguments (4)
00000004 0491 1031 : FSB_DISP = 04 : FSB address
00000008 0491 1032 : VAR_DISP = 08 : value address
0000000C 0491 1033 : FLD_DISP = 12 : field width by value
00000010 0491 1034 : NOT_DISP = 16 : (not used)
0491 1035 :
0491 1036 : Other constants
0491 1037 :
0000000B 0491 1038 : OMAX = 11 : maximum field for leading zeros
0491 1039 :
0000 0491 1040 : .ENTRY PASSWRITEOCTD,^M<>
50 0C AC 0B C3 0493 1041 : SUBL3 #OMAX,FLD_DISP(AP),R0 : R0 = field width low bytes
06 14 0498 1042 : BGTR 110$
50 0C AC D0 049A 1043 : MOVL FLD_DISP(AP),R0
16 11 049E 1044 : BRB 111$
04A0 1045 :
04A0 1046 : Print low order longword
04A0 1047 :
04A0 1048 : 110$:
00 DD 04A0 1049 : PUSHL #0
50 DD 04A2 1050 : PUSHL R0
04 08 AC C1 04A4 1051 : ADDL3 VAR_DISP(AP),#4,R0
60 DD 04A9 1052 : PUSHL (R0) : low order longword
04 AC DD 04AB 1053 : PUSHL FSB_DISP(AP)
FF64 CF 04 FB 04AE 1054 : CALLS #4,PASSWRITEOCT
50 0B D0 04B3 1055 : MOVL #OMAX,R0 : field width high bytes
04B6 1056 :
04B6 1057 : Print R0 digits of high order longword
04B6 1058 :
04B6 1059 : 111$:
00 DD 04B6 1060 : PUSHL #0
50 DD 04B8 1061 : PUSHL R0
08 BC DD 04BA 1062 : PUSHL @VAR_DISP(AP)
04 AC DD 04BD 1063 : PUSHL FSB_DISP(AP)
FF52 CF 04 FB 04C0 1064 : CALLS #4,PASSWRITEOCT
04 04C5 1065 : RET
04C6 1066 :
04C6 1067 :
0000 04C6 1068 : .PSECT _PASSCODE, PIC,EXE,SHR,NOWRT
04C6 1069 :
04C6 1070 : *****
04C6 1071 : * ZERO_FILL_R3 *
04C6 1072 : *
04C6 1073 : *****
04C6 1074 :
04C6 1075 :
04C6 1076 : JSB routine to zero-fill octal and hex output
04C6 1077 :
04C6 1078 :
04C6 1079 : ZERO_FILL_R3:
52 0C AC CE 04C6 1080 : MNEGL FLD_DISP(AP),R2 : entry point
54 52 D1 04CA 1081 : CMPL R2,R4 : get length
03 1B 04CD 1082 : BGEQ $30
```

```

      52 54 DO 04CF 1083
      53 04 BC DO 04D2 1084 $30:
      6342 20 91 04D6 1085 $10:
              OC 12 04DA 1086
      6342 30 90 04DC 1087
EE 52 FFFFFFFF 8F F2 04E0 1088
              05 04E8 1089 $20:
              04E9 1090
              04E9 1091
0000 04E9 1092 .PSECT _PASSCODE, PIC,EXE,SHR,NOWRT
              04E9 1093
              04E9 1094
              04E9 1095
              04E9 1096
              04E9 1097
              04E9 1098
              04E9 1099
              04E9 1100
              04E9 1101
              04E9 1102
              04E9 1103
              04E9 1104
00000004 04E9 1105
00000008 04E9 1106
              04E9 1107
0004 04E9 1108
OC 52 08 AC DO 04EB 1109
A2 04 AC DO 04EF 1110
              04 04F4 1111
              04F5 1112
              04F5 1113
0000 04F5 1114 .PSECT _PASSCODE, PIC,EXE,SHR,NOWRT
              04F5 1115
              04F5 1116
              04F5 1117
              04F5 1118
              04F5 1119
              04F5 1120
              04F5 1121
              04F5 1122
              04F5 1123
              04F5 1124
              04F5 1125
00000004 04F5 1126
              04F5 1127
              04F5 1128
001C 04F5 1129
53 0000005C 8F 04 AC DO 04F7 1130
      54 18 52 C1 04FB 1131
      62 28 A4 D1 0503 1132
              09 13 0507 1133
              52 DD 050B 1134
00000000'EF 01 FB 050D 1135
              00 DD 050F 1136
              01 DD 0516 1137
              01 DD 0516 1138
              01 DD 0518 1139

      MOVL R4,R2
      MOVL @FSB_DISP(AP),R3
      CMPB #SPACE,(R3)[R2]
      BNEQ $20
      MOVB #ZERO,(R3)[R2]
      AOBLSS #-1,R2,$10
      RSB
      ; move address to R3
      ; check next byte for blank
      ; done if not blank
      ; put in zero
      ; return

      *****
      * PASSLINELIMIT *
      *****

      Sets the linelimit for a given file.

      Argument offsets

      AP
      FSB_DISP = 04
      VAL_DISP = 08
      ; number of arguments (2)
      ; FSB address
      ; linelimit value

      .ENTRY PASSLINELIMIT,^M<R2>
      MOVL 8(AP),R2
      MOVL 4(AP),FSB$L_LIM(R2)
      RET

      *****
      * PASSPAGE *
      *****

      Writes a page eject character (1H1 or FORMFEED) to the designated file.

      Arguments offsets

      AP
      FSB_DISP = 04
      ; number of arguments (1)
      ; FSB address

      .ENTRY PASSPAGE,^M<R2,R3,R4>
      MOVL FSB_DISP(AP),R2
      ADDL3 R2,7<FSB$C_BLN+RAB$C_BLN>,R3; R3 = FAB address
      ADDL3 R2,#FSB$C_BLN,R4 ; R4 = RAB address
      CMPL RAB$L_RBF(R4),(R2)
      BEQL 10$
      PUSHL R2
      CALLS #1,PASSWRITELN
      ; terminate current line

      10$:
      PUSHL #0
      PUSHL #1
      ; fill
      ; field width
```

04 1E A3	00	EO	051A	1140	BBS	#FABSV_FTN,FAB\$B_RAT(R3),20\$; check for carriage control
	0C	DD	051F	1141	PUSHL	#FORMFEED ; not FORTRAN
	02	11	0521	1142	BRB	30\$
			0523	1143		
	31	DD	0523	1144	20\$:	
			0525	1145	30\$:	
	52	DD	0525	1146	PUSHL	#ONE ; FORTRAN
FB2F CF	04	FB	0527	1147	PUSHL	R2 ; FSB address
	52	DD	052C	1148	CALLS	#4,PASSWRITECHAR
00000000'EF	01	FB	052E	1149	PUSHL	R2
		04	0535	1150	CALLS	#1,PASSWRITELN ; terminate line
			0536	1151	RET	; return
			0536	1152		
			0536	1153		
			0536	1154		
					.END	

PASSIO_OUTPUT
Symbol table

; PASCAL RMS linkage

G 2

16-SEP-1984 02:07:46 VAX/VMS Macro V04-00
5-SEP-1984 02:32:22 [PASCAL.SRC]PASIO3.MAR;1

Page 22
(1)

```

$$TMP1      = 00000001
$$TMP2      = 00000057
$10         = 000004D6 R      02
$20         = 000004E8 R R    02
$25         = 000003A3 R R    02
$30         = 000004D2 R R    02
$35         = 000003D8 R R    02
$40         = 000003E1 R R    02
$43         = 00000490 R R    02
$55         = 00000452 R R    02
$65         = 00000487 R      02
CHR_DISP    = 00000008
DIG_DISP    = 00000010
DOB_DISP    = 00000008
DSCSA_POINTER = 00000004
DSCSC_S_BLN = 00000008
DSCSW_LENGTH = 00000000
EMIN        = 00000008
FABSB_FNS   = 00000034
FABSB_RAT   = 0000001E
FABSL_FNA   = 0000002C
FABSV_FTN   = 00000000
FLD_DISP    = 0000000C
FMAX        = 0000002A
FMIN        = 00000003
FOR$CNV_OUT_D = ***** X      00
FOR$CNV_OUT_E = ***** X      00
FOR$CNV_OUT_F = ***** X      00
FOR$CNV_OUT_I = ***** X      00
FOR$CNV_OUT_O = ***** X      00
FOR$CNV_OUT_Z = ***** X      00
FORMFEED    = 0000000C
FSBSC_BLN   = 00000018
FSBSL_LIM   = 0000000C
FSBSL_LST   = 00000008
FSB_DISP    = 00000004
HMAX        = 00000008
IMAX        = 00000014
IMINN       = 00000002
IMINP       = 00000001
INT_DISP    = 00000008
LEN_DISP    = 00000010
MAX_DISP    = 00000014
NAMELEN     = 00000020
NAM_DISP    = 00000010
NEWENT      = 0000000D R      02
NOT_DISP    = 00000010
OMAX        = 0000000B
ONE         = 00000031
OVERFLOWSIZE = 00000014
PASSBUFFEROVER = ***** X      00
PASSIOERROR = ***** X      00
PASSLINELIMIT = 000004E9 RG    02
PASSPAGE    = 000004F5 RG    02
PASSPUTBIN   = 00000000 RG    02
PASSPUTBINARY = 0000000B RG    02
PASSPUTTXT   = 0000003A RG    02

```

```

PASSWREAL   = 00000219 R      02
PASSWREALF  = 000002A7 R      02
PASSWRITECHAR = 0000005B RG    02
PASSWRITEDOUBE = 000001F6 RG    02
PASSWRITEDOUBF = 0000028A RG    02
PASSWRITEHEX  = 00000368 RG    02
PASSWRITEHEXD = 000003E2 RG    02
PASSWRITEINT  = 0000014C RG    02
PASSWRITELN   = ***** X      00
PASSWRITEOCT  = 00000417 RG    02
PASSWRITEOCTD = 00000491 RG    02
PASSWRITEOK   = ***** X      00
PASSWRITEREAL = 00000209 RG    02
PASSWRITEREALF = 0000029A RG    02
PASSWRITESCAL = 000000F2 RG    02
PASSWRITESTR  = 00000098 RG    02
RABSB_RAC     = 0000001E
RABSC_BLN     = 00000044
RABSC_SEQ     = 00000000
RABSL_RBF     = 00000028
RABSL_ROP     = 00000004
RABSM_TPT     = 00000002
REL_DISP      = 00000008
SCA_DISP      = 00000008
SPACE         = 00000020
STAR          = 0000002A
STR_DISP      = 00000008
SYSPUT        = ***** G      02
VAL_DISP      = 00000008
VAR_DISP      = 00000008
ZERO          = 00000030
ZERO_FILL_R3  = 000004C6 R      02

```

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR
\$ABS\$	00000000 (0.)	01 (1.)	NOPIC USR
_PAS\$CODE	00000536 (1334.)	02 (2.)	PIC USR

CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE
CON	ABS	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
CON	REL	LCL	SHR	EXE	RD	NOWRT	NOVEC	BYTE

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	34	00:00:00.08	00:00:00.50
Command processing	135	00:00:00.45	00:00:01.24
Pass 1	252	00:00:08.14	00:00:16.17
Symbol table sort	0	00:00:00.87	00:00:00.91
Pass 2	194	00:00:02.86	00:00:04.53
Symbol table output	12	00:00:00.07	00:00:00.09
Psect synopsis output	3	00:00:00.03	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	633	00:00:12.50	00:00:23.47

The working set limit was 1500 pages.

49931 bytes (98 pages) of virtual memory were used to buffer the intermediate code.

There were 40 pages of symbol table space allocated to hold 726 non-local and 40 local symbols.

1154 source lines were read in Pass 1, producing 61 object records in Pass 2.

14 pages of virtual memory were used to define 12 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	9

772 GETS were required to define 9 macros.

There were no errors, warnings or information messages.

MACRO/DISABLE=TRACE/LIS=LIS\$:PASIO3/OBJ=OBJ\$:PASIO3 MSRC\$:PASIO3/UPDATE=(ENH\$:PASIO3)

0292 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

0293 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

